A (Practical Training Report)on

CHESS GAME

Under The Guidance Of
Deepthi and Sangeetha Ma'am

Submitted By:

BHAVYA BORDIA - 16IT212

BHARAT SHARMA - 16IT210

ADESH SANGOI - 16IT102

Anmol Jindal – 16IT222

SHUBHAM RAJ - 16IT139

3RD SEMESTER B-TECH (IT)

In partial fulfillment in award of the degree

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY
National Institute of Technology Karnataka,Surathkal

DECLARATION

We hereby declare that the unix project work entitled "Chess Game" submitted to the National Institute of Technology Karnataka, is a record of an original work plus with the help of the available resources is done by us under the guidance of Sangeetha Ma'am and Deepti Ma'am**,** Of Department of Information Technology, National Institute of Karnataka and this project work is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Name of the group members:

BHAVYA BORDIA - 16IT212

BHARAT SHARMA - 16IT210

ADESH SANGOI - 16IT102

ANMOL JINDAL - 16IT222

SHUBHAM RAJ - 16IT139

Project Abstract

CHESS

This project implements a classic version of Chess with a Graphical User Interface. The Chess game follows the basic rules of chess, and all the chess pieces only move according to valid moves for that piece. Our implementation of Chess is for two players (no Artificial Intelligence). It is played on an 8x8 checkered board, with a dark square in each player's lower left corner. Initially developing a text-based version, we then used Unicode characters to implemented it in a GUI. Although we were not able to design a working artificial intelligence for the chess game in the time allowed, we researched the past implementations. We successfully created a GUI using inheritance and templates, as specified. Despite several unusual bugs in the GUI, our Chess program is a great, user-friendly game for two players.
The project involves basic and advanced application of shell script and unix commands to enhance our understanding in the course.

Index

List of figures

INTRODUCTION

This project implements a classic version of Chess using shell script and a Graphical User Interface and this version of chess runs on the terminal as normal .sh file. The Chess game follows the basic rules of chess, and all the chess pieces only move according to valid moves for that piece. Our implementation of Chess is for two players (no Artificial Intelligence). It is played on an 8x8 Checkered board, with a dark square in each player's upper left corner of the terminal.

Below is one of the screenshot the GUI that we created using Unicode.

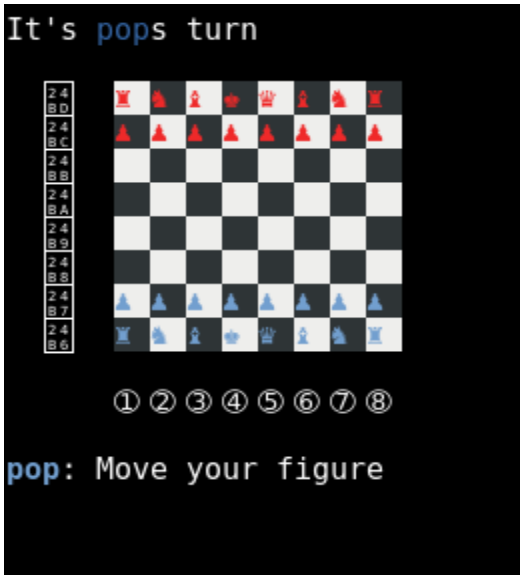

Figure 1

CHALLENGES FACED

- Creating GUI for the chessboard like king, Queen, bishop, knight etc.
- Creating a local remote server to run two player game.
- Taking input via cursor i.e tracking the movement of the cursor on the terminal to track its coordinates where it had clicked
- Checking the validity of all the correct moves which had been done by the users playing the game and displaying an appropriate message.
- Delete cache memory which is formed in the game and displaying the moves while playing the game.

MOTIVATION TO DO WORK

- To enjoy in the spare time and use the time to enhance our thinking skill.
- As it a terminal based game we can play it in our labs also
- To enhance our coding skill in the shell script.
- Carrying a custom of formality two players square of against one another a game is therefore not only one of the tactics but also psyche.

METHODOLOGY AND FRAMEWORK

We started by planning a flowchart for our chess game, and constructing a checkerboard in the starting so that we can have the basic layout . This organized our plan for coding. Next we began building up the GUI using the Unicode characters and bash Ansi colors for this version of the Chess game and we togetherly started testing it by doing brainstorming and using the available resource that we have. Next we started the planning of design for the logic we should use Keeping in mind that all the rules of the game should be followed. After getting success in the above we go for the cursor movement from the mouse side to make the chess game more user friendly and we also check who win/losses the game and display the winner in it.
As it is a two player game so two player will always be required to play it efficiently.

SYSTEM REQUIREMENTS

- A Linux based operating system.
- Bash compatible system.
- With preinstalled Unicode library to run the effectively(optional)
- System should be capable of making remote server.
- Capable of detecting cursors clicks

ALGORITHM AND TECHNIQUES

First of all we followed basic rules of chess which are as mentioned below:

- Each chess piece has its own style of moving.
- The Xs mark the squares where the piece can move if no other pieces (including one's own piece) are on the Xs between the piece's initial position and its destination.
- If there is an opponent's piece at the destination square, then the moving piece can capture the opponent's piece.
- The only exception is the pawn which can only capture pieces diagonally forward.

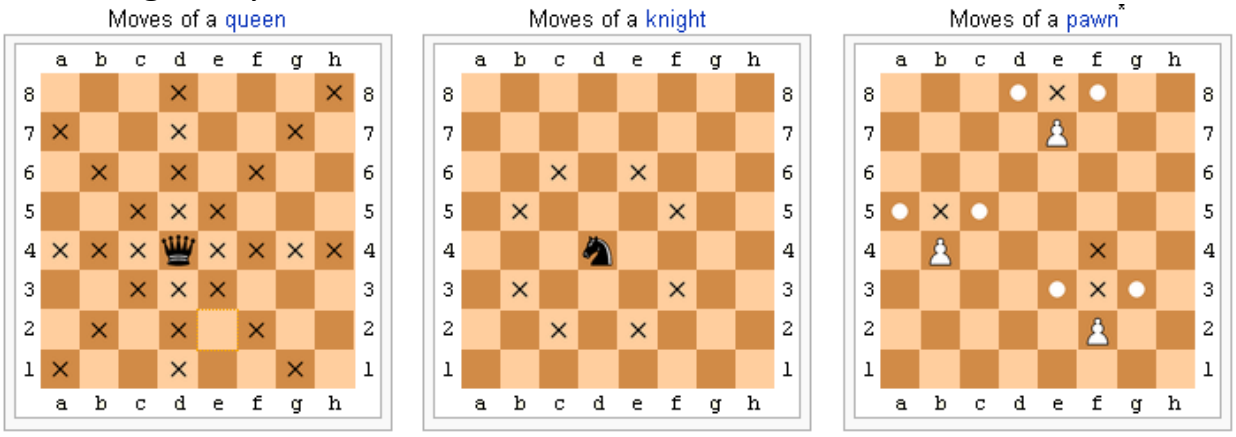

Figure 2

- When a pawn advances to its eighth rank, it is exchanged for the player's choice of a queen, rook, bishop, or knight of the same color.
- Usually, the pawn is chosen to be promoted to a queen

Also we had kept in mind that the starting move will be of the blue color side only in our game

Thing that we have also consider used techniques while coding the game:

- We have assigned positive values for 1 player and negative for the other player and the empty slots are assigned a value zero.
- So every warrior of the player one will have the positive value and every warrior of the player two will have negative value so that we can keep check on them.
- Also we have assigned a priority number to identify the behavior of the warrior on each side
  For example priority Five is given to the queen, as shown in the image below:

```
# readable figure names
declare -a figNames=( "(empty)" "pawn" "knight" "bishop" "rook" "queen" "king" )
# ascii figure names (for ascii output)
declare -a asciiNames=( "k" "q" "r" "b" "n" "p" " " "P" "N" "B" "R" "Q" "K" )
# figure weight (for heuristic)
declare -a figValues=( 0 1 5 5 6 17 42 )
```

Figure 3

- We have used the method of hasKing() to check whether the game has ended or not like if there is no king anyone of the player on the board means that the opposite player won and the game has ended.

- We defined a function canmove which follow all the rules of chess game which are metioned in the starting to check whether the move done by the player is a valid one or not and display the appropriate message on the terminal.
- We have used the Unicode inbuilt characters to form the checkerboard and the ansi colors also as illustrated in the code

You can find from there a screenshot suggested that part attached below:

```
function drawField(){
  local y=$1
  local x=$2
  echo -en "\e[0m"
  # move coursor to absolute position
  if $3 ; then
    local yScr=$(( y + originY ))
    local xScr=$(( x * 2 + originX ))
    if $ascii && (( x >= 0 )) ; then
      local xScr=$(( x * 3 + originX ))
    fi
    echo -en "\e[${yScr};${xScr}H"
  fi
  # draw vertical labels
  if (( x==labelX && y >= 0 && y < 8)) ; then
    if $hoverInit && (( hoverY == y )) ; then
      if $color ; then
        echo -en "\e[3${colorHover}m"
      else
        echo -en "\e[4m"
      fi
    elif (( selectedY == y )) ; then
      if ! $color ; then
        echo -en "\e[2m"
      elif (( ${field[$selectedY,$selectedX]} < 0 )) ; then
        echo -en "\e[3${colorPlayerA}m"
```

Figure 4

- We have used the function inputcoord function that will take the input of the coordinates selected using the mouse from the terminal and give the coordinates to make us know where we have to move and the we check the validity of the move using can move and we update the field in order to show the updated move. The snippet of the code is attached below:

```
function input() {
  local player=$1
  SECONDS=0
  message="\e[1m$(namePlayer "$player")\e[0m: Move your figure"
  while true ; do
    selectedY=-1
    selectedX=-1
    title="It's $(namePlayer "$player")s turn"
    draw >&3              #Draw battlefield
    if inputCoord ; then
      selectedY=$inputY
      selectedX=$inputX
      if (( ${field["$selectedY,$selectedX"]} == 0 )) ; then
        warn "You cannot choose an empty field!" >&3
      elif (( ${field["$selectedY,$selectedX"]} * player  < 0 )) ; then
        warn "You cannot choose your enemies figures!" >&3
      else
        send "$player" "$selectedY" "$selectedX"          #Moving to check can move. Wait for next corrdinate
        local figName=$(nameFigure ${field[$selectedY,$selectedX]} )
        message="\e[1m$(namePlayer "$player")\e[0m: Move your \e[3m$figName\e[0m at $(coord "$selectedY" "$selectedX")
        draw >&3
        if inputCoord ; then
          selectedNewY=$inputY
          selectedNewX=$inputX
          if (( selectedNewY == selectedY && selectedNewX == selectedX )) ; then
            warn "You didn't move..." >&3
```

Figure 5

- Use of local server to host a two player game on our system which is predefined code .

  Using this concepts and techniques we have coded in the shell programing language and make our project successful.

IMPLEMENTATION

We have implements all the techniques what we have mentioned in the techniques and the algorithm part.

Some of the unix commands that we have used in our code are :

1.) Type
2.) Read –sN1 a
3.) Getopts command of shell programming
4.) nc –l
5.) echo –en "/e[34m"

Syntax:

- **Type Command:**

The type command is used to find out if command is builtin or external binary file. It also indicate how it would be interpreted if used as a command name.

Find Out Command Type (-t option)

If the -t option is used, it will print a single word which is one of the following

- alias (command is shell alias)
- keyword (command is shell reserved word)
- function (command is shell function)
- builtin (command is shell builtin)
- file (command is disk file)

type -t Command Examples

Try the following examples:

Command	Output	Meaning
type -t ls	alias	ls command is alias which can be verified by typing the alias command itself at a shell prompt: `alias`
type -t date	file	date command is a disk file (external command), which can be verified by issued the **which date** command at a shell prompt: `which date`
type -t xrpm	function	xrpm is a user defined function.
type -t if	keyword	if is a shell reserved word, which is used for flow control.
type -t pwd	builtin	pwd is a shell builtin command.

Return Command Name Type (-p option)

The -p option is used to find of the name of the disk file (external command) would be executed by the shell. It will return nothing if it is not a disk file.

We used stty along with type to disable the echoing on the screen

Command :

Stty –echo #this will diable the visibility
And to get back to original state
Stty echo

#this will bring back the visibility on the screen

- **NC command**

Nc stands for netcat. The simple unix utility which reads and writes data across netwok connection using tcp Protocol.

Command which we have used in this we have studied:
Syntax :Nc –l port

-l stands for listenport
And port is the port number
To check the connection is stablished between the post or not

Syntax : Nc hostname port
Hostname will carry hostip
System will be connected to the host using port number we have given

- Read –SN1 a:
This command will read the 1 character only which is displayed on the screen and store it in a variable we have used this for getting position of mouse

- echo –en '\e[18t'
  This is the command which we used to get the terminal dimension of the screen
- echo –en '\e[40m'
  This command is used while building up the checker board this is the command which we used to extract the ansi colors
- for enabling the mouse cursor it:
  echo –en '\e[?9h'
  this will be used to get the cursor coordinates via mouse

- we have use getopts command:

Usage :
getopts [optsagruments]

we have used in the following way:

```
# Parse command line arguments
while getopts ":a:b:h" options; do
  case $options in
    a ) namePlayerA="$OPTARG";;
    b ) namePlayerB="$OPTARG"
      ;;
    h ) help
      exit 0
      ;;
    \?)
      echo "Invalid option: -$OPTARG" >&2
      ;;
  esac
done
```

Figure 6

TESTING AND ANALYSIS

- Move must be alternate i.e. no player can skip his/her move.
- Checking available moves.
- Checking moves on corner cases.
- Checking availability of both king to continue the game.
- Satisfying Pawn to Queen Transition.
- Finally by hosting a game between two of the best players of our department.
- Indefinite number of combinations of moves are possible in a chess game.
- Shell provides Quick start, and interactive debugging.

RESULT

Successful implementation of the chess game in shell script some of the screen shot of working game are as shown:

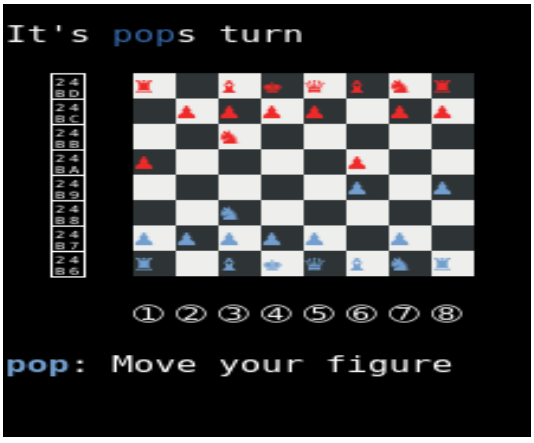

Figure 7

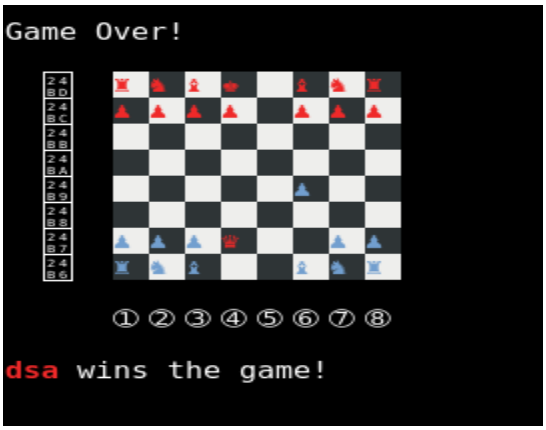

Figure 8

CONCLUSION

In this project we presented the classic version of Chess implemented as a GUI. We successfully created a game using shell scripting, and a GUI (and learned how to use Bash ascii colors and different unix commands in the process). Although we encountered some minor bugs, we were able to design a working, user-friendly graphical interface for our Chess game.

REFERENCES

- Concepts and Applications Unix ,Sumitab Das
- For GUI:
  http://www.utf8-chartable.de/unicode-utf8-table.pl?start=9728&number=128&names=-
- For ANSI Bash colors :
  https://gist.github.com/inexorabletash/9122583
- https://en.wikipedia.org/wiki/Rules_of_chess
- Seniors Help :
  THIRD YEAR SENIORS , IT DEPT , NITK.